APPENDIX D

Subsystem Interface

```
public interface IModelService
{
        // set the configuration object used to configure the behavior of ModelService
        void SetConfig(Config conifg);

        // add an xml ER-map file to be converted
        void AddMapFile(string fileName);

        // tell ModelService to start processing
        void Process();
}

public interface IMapLoader
{
        // tell MapLoader to use a particular XSLT file for transforming the XML file
before conversion to EntityMap
        void SetMapTransformFile(string mapTransformFileName);

        // add a map file to the list of map files to be loaded
        void AddMapFile(string fileName);

        // tell MapLoader to load all the map files added
        void LoadMaps();

        // retrieve the output collection of entity maps loaded
        EntityMapCollection EntityMaps { get; }
}

public interface IMapWalker
{
        // set the schema name to be used in the DataTable entries
        void SetDBSchemaName(string dbSchemaName);

        // set a collection of entity maps to be walked
        void SetEntityMapCollection(EntityMapCollection entityMaps);

        // set the measure hints
        void SetMeasureHints(MeasureHintCollection measureHints);

        // walk the entity maps
        void WalkEntityMaps();
```

```
        // retrieve the resulting dataset schema generated
        DataSet Schema { get; }
}

public interface IModelGenerator
{
        // set the connect string of the default data source
        void SetDataSource(string dbServerName, string dbDatabaseName);

        // set the hint object to be used during model generation
        void SetHint(Hint hint);

        // set the dataset schema to be processed
        void SetSchema(DataSet dataset);

        // generate the UDM Model from the supplied dataset schema
        void Generate();

        // retrieve the resulting model generate
        UDMModel UdmModel { get; }
}

public interface IModelMaterializer
{
        // set the UDM server to use for the materialization
        void SetUDMServerName(string udmServerName);

        // set the log file for doing log only
        void SetLogFile(string logFileName);

        // instruct ModelMaterializer to drop other UDM databases before materializing
        void SetDropAllDatabases(bool drop);

        // set the model to be materialized
        void SetUdmModel(UDMModel udmModel);

        // materialize a UDM model onto the UDM server
        void Materialize();

        // process a previously materialized UDM Model
        void Process();
}
```

```
public interface ICodeGenerator
{
        // set the generator to be invoked for the real work
        void SetBICodeGenerator(IBIGenerator generator);

        // set the UDM Model whose code is to be generated from
        void SetUdmModel(UDMModel udmModel);

        // start code generation for the given model
        void Generate();
}
```